**Chapter 2 Selection - an Introduction**

The program below illustrates the simplest **decision-making** (or **selection**) process in programming - it uses the **IF** statement:-

*// A simple program to demonstrate the simple IF statement*

*// Chap0201.cpp*

#include <iostream>

using namespace std;

void main()

{

int age;

cout << "Retirement calculator" << endl << endl;

cout << "Type in your age:- ";

cin >> age;

if (age == 65)

cout << endl << "You may retire this year";

cout << endl << endl;

system("pause");

}

Notice the following:-

- the IF statement starts with the word **if** and ends at the first semi-colon. There is no semi-colon at the end of the line with the word **if**

- the body of the IF statement is indented. This is a convention and is good programming style - though it has no effect on the operation of the program

- the operator to test for equality is '**==**' and not '**=**' ('**=**' is an assignment operator in C and C++)

- the Boolean expression '**age** **==** **65**' must be enclosed in brackets. We say it is a Boolean expression because its value can only be true or false.

The six **comparison operators** which may be used in Boolean expressions are:-

**==** is equal to !**=** is not equal to

**>** is greater than **>=** is greater than or equal to (not less than)

**<** is less than **<=** is less than or equal to (not greater than)

The next program illustrates the use of another of these comparison operators:-

*// Program to show the use of the > (greater than) operator*

*// Chap0202.cpp*

#include <iostream>

using namespace std;

void main()

{

int age;

cout << "Retirement calculator" << endl << endl;

cout << "Type in your age:- ";

cin >> age;

if (age > 65)

cout << endl << "You should have retired by now";

cout << endl << endl;

system("pause");

}

Now try modifying the above program so that it makes an appropriate response if you type in an age which is less than 65. Spend some time trying out the other comparison operators in your program to get the feel of them**.**

The next program introduces a little more complexity into the IF statement:-

// Program to showe the IF statement with an ELSE clause

// Chap0202.cpp

**#include <iostream>**

using namespace std;

**void main()**

**{**

**int first, second;**

**cout << "Division calculator" << endl << endl;**

// Get two numbers

**cout << "Type in an integer:- ";**

**cin >> first;**

**cout << "Type in another:- ";**

**cin >> second;**

// Decide if division is possible

**if (second == 0)**

**cout << endl << "Can't do the division";**

**else**

**cout << endl << "Could do the division";**

**cout << endl << endl;**

system("pause");

**}**

Note that the first '**cout**' statement is the action to be executed if the value of the Boolean expression '**second** **==** **0**' is **true** - the second '**cout**' statement is the **false** action. It is not possible for both actions to be carried out – because the Boolean conditional test must have come out to true or false.

You will often want a program to do two or more things during either the true or false part of an IF statement. The next program illustrates how to accomplish this:-

// Demonstrates an IF statement with several ELSE actions

// Chap0204.cpp

**#include <iostream>**

using namespace std;

**void main()**

**{**

**int first, second;**

**float quotient;**

**cout << "Division calculator" << endl << endl;**

// Get two numbers

**cout << "Type in an integer:- ";**

**cin >> first;**

**cout << "Type in another:- ";**

**cin >> second;**

// Divide first by second if possible and display result

**if (second == 0)**

**cout << "Can't do the division";**

**else**

**{**

**quotient = (float)first / second;**

**cout.precision(6);**

**cout << first << " / " << second << " = " << quotient;**

**}**

**cout << endl << endl;**

system("pause");

**}**

The above program illustrates what must be done if either the true or the false action has to use more than a single statement. A **compound statement** must be formed by using braces **{..}**.

Add a second '**cout**' statement to the true outcome of the IF statement which beeps at the user and says what the problem is ('can't divide by zero').

***CHAR variables***

We have already met int and float variables. A third type of variable is the char. A char can store any single character. The character can be alphabetic or numeric or a special character. A char is said to be alphameric.

Here’s a demo of the char variable:-

// Char demo

// PY Oct 2013

#include <iostream>

using namespace std;

void main()

{

char inGender;

cout << "Please enter your gender (M or F) - ";

cin >> inGender;

if (inGender == 'M')

cout << "You are male" << endl << endl;

else

cout << "You are female" << endl << endl;

system("pause");

}

Note that in C++ we use single quotes ' for one character and double quotes " for text strings.

**Complex Ifs – using AND and OR**

Here we will introduce the use of the ***logical operators*** AND and OR and some of the uses to which they might be put in your programs:-

*// This program demonstrates the use of the AND (&&) operator*

*// It checks whether an input is from 3 to 7 inclusive*

*// Chap0206.cpp*

**…**

**void main()**

**{**

**int inNum;**

**cout << "Range checking" << endl << endl;**

**cout << "Enter a number:- ";**

**cin >> inNum;**

**if ( inNum > 2 && inNum < 8)**

**cout << endl << "Number is in range";**

**else**

**cout << endl << "Number is out of range";**

**…**

**}**

This program checks whether a number entered by the user is within a certain range. It uses, in a boolean expression, the AND operator which is written '**&&**' in C++. Be very careful to type two ampersands - because a single ampersand is a 'bitwise AND operator' which will not work as you expect!

Note that you cannot say in the boolean expression:-

**if (inNum > 2 && < 8)** *// This is* ***incorrect***

We can use common sense to understand this shorthand version. But it makes no sense at all to C++ because the composite boolean expression should consist of two simpler complete boolean expressions:-

inNum > 2 and inNum < 8

Can you determine from an examination of the code what the boundaries of the range are? Check your answer by thorough testing of the program.

Or we could use the more professional approach of setting up named constants for the range boundaries.

There is another slight change now to the checking of the boundaries. Can you see what it is - and what effect it has on the range?

*// This program demonstrates better technique in setting up a*

*// range check using pre-defined constants*

*// Chap0502.cpp*

**…**

**#define MIN 0**

**#define MAX 100**

**void main()**

**{**

**int inNum;**

**cout << "Range checking" << endl << endl;**

**cout << "Enter a number:- ";**

**cin >> inNum;**

**if (inNum >= MIN && inNum <= MAX)**

**cout << endl << "Number is in range ";**

**else**

**cout << endl << "Number is out of range";**

**…**

**}**

The next program uses the OR (||) operator to detect the presence of specific values - in this case an 'A' or an 'S'. Again note the double vertical bar - a single one is a bitwise OR operator. The double vertical bar is often referred to as a 'pipe' symbol.

*// This program demonstrates the use of the OR (||) operator*

*// Chap0502.cpp*

**…**

**void main()**

**{**

**char reply;**

**cout << "Maths choice" << endl << endl;**

**cout << "Choose A for adding or S for subtracting:- ";**

**cin >> reply;**

**if (reply == 'A' || reply == 'S')**

**cout << endl << "Acceptable reply";**

**else**

**cout << endl << "You must type 'A' or 'S'";**

**…**

**}**

This program uses the OR operator in another valid approach to range checking:-

*// This program uses OR in range checking*

*// Chap0207.cpp*

**…**

**void main()**

**{**

**int inNum;**

**cout << "Range checking" << endl << endl;**

**cout << "Enter a number:- ";**

**cin >> inNum;**

**if (inNum < 2 || inNum > 8)**

**cout << endl << "Number is out of range";**

**else**

**cout << endl << "Number is in range";**

**…**

**}**

**IF exercises**

**Test each program thoroughly** before moving on.

2.01 Design and write a program which prompts the user for a bank balance figure. If the balance is over £1000 then output a message saying:- “You are rich”.

*XXIF01.CPP*

2.02 Design and write a program similar to 2.01 but also outputting an alternative message to users who are not ‘rich’ – i.e. you will have to use IF .. ELSE.

*XXIF02.CPP*

2.03 Design and write a program which asks the user for a height in metres. If it is at least 2 metres then output a message saying:- “You are very tall”.

*XXIF03.CPP*

2.04 Design and write a program similar to 2.03 but also outputting a message to students who are not 2 metres tall telling them they are ‘not very tall’.

*XXIF04.CPP*

2.05 Design and write a program similar to 2.03 but outputting two possible messages. Tell people reaching 2 metres that they are ‘very tall’; tell people below 1.80 metres that metres that they are ‘not tall’ – i.e. use two separate IFs.

NB People between these two bands should not see any messages.

*XXIF05.CPP*

2.06 Design and write a program similar to 2.05 but outputting three possible messages. Tell people of at least 2 metres that they are ‘very tall’; tell people below 1.80 metres that they are ‘not tall’; and tell people in between (i.e. 1.80 metres to 1.99 metres) that they are ‘quite tall’.

NB Be careful that **everybody sees one message** and only one message

*XXIF06.CPP*

2.07 Design and write a program to receive a sum in £ from the user. Then ask the user if VAT is payable. If it is payable calculate and add the VAT before outputting the final amount. Otherwise just output the total without adding VAT.

*XXIF07.CPP*

2.08 Design and write a program to receive a sum in £ from the user. Then ask whether the money is to be converted into $ or into €. Calculate and output the appropriate converted amount. Use £1 = $1.50 = €1.20

*XXIF08.CPP*

2.09 As exercise 2.08 above but prompt the user initially for both of the exchange rates for today (£ to $ and £ to €).

*XXIF09.CPP*

2.10 As exercise 2.09 above but after the initial entry of exchange rates make the script perform three conversions before closing – i.e. not just one conversion.

*XXIF10.CPP*

2.11 Design and create a C++ program to handle on-line applications for the post of Security Guard at the famous P In The Dark rock festival.

The **rules** for approving an application are as follows:-

* Males must be at least 1.8 metres tall
* Females must be at least 1.6 metres tall
* Everyone must be aged 18 or over

End each application with **one** of these two messages:-

* "You are eligible"
* "You are not eligible"

**Hints**:

* **How many questions** do you have to ask in total?
* Which one should you ask **first**?

Make a test log like this and use it to test that you have covered all the possibilities correctly:-

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Age | Height | M/F | Expected | Actual | Comment |
| 17 | 2 | M | No |  |  |
| 17 | 1.5 | F | No |  |  |
| 18 | 2 | M | Yes |  |  |
| 18 | 1.5 | M | No |  |  |
| 19 | 1.7 | M | No |  |  |
| 19 | 2 | F | Yes |  |  |
| 19 | 1.5 | F | No |  |  |
| 19 | 1.7 | F | Yes |  |  |
| … | … | … | … |  |  |

2.12 Employees earn at the rate of £7 per hour if they work 37.5 hours or less during the week. For any hours over 37.5 they are paid overtime at £8.50 per hour. Write a program which asks how many hours an employee has worked and which then works out the employee’s gross pay. You can ignore tax etc.

Notice that your solution will require careful testing. Unless you are absolutely brilliant at maths you won’t be able to input a number of hours and know straight away whether the computer’s answer is correct.

Again you should compile a list of test data in advance and work out the correct gross pay for every number of hours you enter. Your test data might include:- 30 hours, 40 hours, 50 hours, 60 hours, 70 hours. First you work out your expected answers. Then you run your program to see whether your design is correct and the actual results match your predicted ones.

*XXIF12.cpp*

2.13Modify the above program (2.12) so that if the employee has worked 60 hours or more they get £10 per hour for every hour beyond 60 hours - in addition to the amounts quoted above.

*XXIF13.cpp*